A Framework for Resilient Remote Monitoring

Michael Atighetchi, Aaron Adler
Raytheon BBN Technologies
Cambridge, MA
{matighet, aadler}@bbn.com

***Abstract*— Today's activities in cyber space are more connected than ever before, driven by the ability to dynamically interact and share information with a changing set of partners over a wide variety of networks. To support dynamic sharing, computer systems and network are stood up on a continuous basis to support changing mission critical functionality. However, configuration of these systems remains a manual activity, with misconfigurations staying undetected for extended periods, unneeded systems remaining in place long after they are needed, and systems not getting updated to include the latest protections against vulnerabilities. This environment provides a rich environment for targeted cyber attacks that remain undetected for weeks to months and pose a serious national security threat. To counter this threat, technologies have started to emerge to provide continuous monitoring across any network-attached device for the purpose of increasing resiliency by virtue of identifying and then mitigating targeted attacks. For these technologies to be effective, it is of utmost importance to avoid any inadvertent increase in the attack surface of the monitored system. This paper describes the security architecture of *Gestalt*, a next-generation cyber information management platform that aims to increase resiliency by providing ready and secure access to granular cyber event data available across a network. Gestalt's federated monitoring architecture is based on the principles of strong isolation, least-privilege policies, defense-in-depth, crypto-strong authentication and encryption, and self-regeneration. Remote monitoring functionality is achieved through an orchestrated workflow across a distributed set of components, linked via a specialized secure communication protocol, that together enable unified access to cyber observables in a secure and resilient way.**

Keywords: cyber security, federated access, semantic web, middleware

I. INTRODUCTION

System administrators and cyber defenders continue to face challenges in securing systems in enterprise environments as attacks keep increasing in the level of sophistication and as the number of connected systems keeps increasing. To support and automate manual activities associated with obtaining information about systems and taking corrective action in response to suspicious activities, an increasing number of technologies for remote monitoring are becoming available with the premise of increasing resiliency by decreasing the time-to-detect and time-to-mitigate targeted attacks.

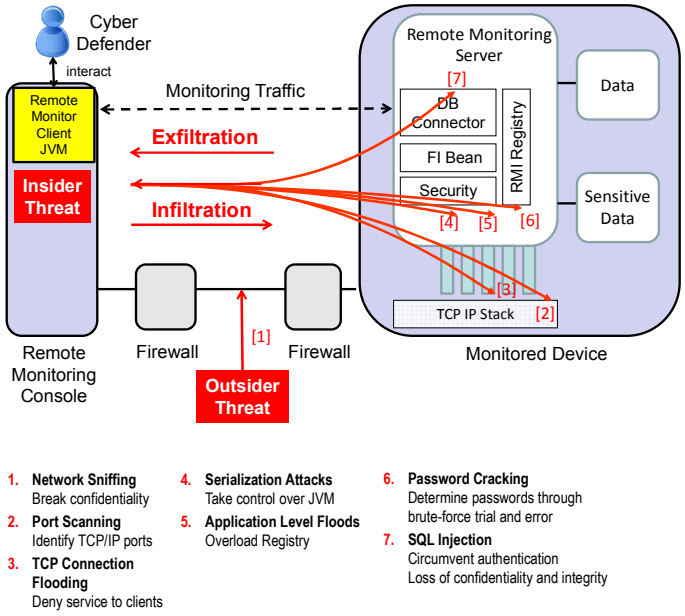

Fig. 1. Attack Surface of an Exemplar Remote Monitoring System

While the functional benefit of new protocols and tools that support continuous monitoring and incident response is clear, it is quite common for these tools to fail on the security front by either (1) providing inadequate security, e.g., by adding to the attack surface and thereby enabling adversaries to remotely monitor / manage critical infrastructure or (2) requiring a very stringent set of security controls that are prohibitively difficult to implement, thereby limiting adopting in the market place. Versions 1 and 2 of the Simple Network Management Protocol (SNMP) [1] provide inadequate security and are widely adopted, while version 3 has started to provide acceptable security but has a limited deployment footprint. WS-Security [2] and the Common Secure Interoperability Protocol Version 2 (CSIv2) [3] are examples of protocols that started out with complex security controls and did not achieve anticipated market penetration and adoption.

As a motivating example, consider an attack progression in a remote monitoring system built on a web services stack as displayed in Fig. 1. The base system consists of a client application (left) interacting with an application server (right), which in turn accesses internal state of the monitored device.

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). This work was supported by the US Defense Advanced Research Project Agency (DARPA) under contract FA8750-14-C-0028. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 2014		**00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Framework for Resilient Remote Monitoring	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon BBN Technologies,10 Moulton Street,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
2nd International Symposium on Resilient Cyber Systems, August 19-21, 2014, Denver, CO. U.S. Government or Federal Rights License

14. ABSTRACT
Today?s activities in cyber space are more connected than ever before, driven by the ability to dynamically interact and share information with a changing set of partners over a wide variety of networks. To support dynamic sharing, computer systems and network are stood up on a continuous basis to support changing mission critical functionality. However, configuration of these systems remains a manual activity, with misconfigurations staying undetected for extended periods, unneeded systems remaining in place long after they are needed, and systems not getting updated to include the latest protections against vulnerabilities. This environment provides a rich environment for targeted cyber attacks that remain undetected for weeks to months and pose a serious national security threat. To counter this threat, technologies have started to emerge to provide continuous monitoring across any network-attached device for the purpose of increasing resiliency by virtue of identifying and then mitigating targeted attacks. For these technologies to be effective it is of utmost importance to avoid any inadvertent increase in the attack surface of the monitored system. This paper describes the security architecture of Gestalt, a next-generation cyber information management platform that aims to increase resiliency by providing ready and secure access to granular cyber event data available across a network. Gestalt?s federated monitoring architecture is based on the principles of strong isolation, leastprivilege policies, defense-in-depth, crypto-strong authentication and encryption, and self-regeneration. Remote monitoring functionality is achieved through an orchestrated workflow across a distributed set of components, linked via a specialized secure communication protocol, that together enable unified access to cyber observables in a secure and resilient way.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

An adversary can start out by gathering sensitive information through network sniffing (attack 1) and perform target identification through port scanning and remote operating system (OS) fingerprinting (attack 2). Once the list of open ports and OS type is determined, a variety of off-the-shelf attacks, such as SYN flooding [4] can be employed against older versions of operating systems to cause denial of service or privilege escalation on systems that have not been hardened or patched appropriately. Advancing up the network stack, the adversary can attack the TCP connection handling code of the event listeners (attack 3) by establishing and maintaining a large number of TCP connections [5], causing thread resource bottlenecks and denying service to legitimate clients. A more sophisticated adversary can attack the Java Virtual Machines (JVMs) on which the services and applications are running (attack 4), e.g., by creating maliciously crafted serialized objects to execute arbitrary code during de-serialization [6]. Similarly, creating specific XML documents can cause SOAP processors to crash. Furthermore, attackers may devise application-level attacks, such as floods targeted at registry services to deny access to legitimate clients by overloading registry processing (attack 5), brute-force password combinations (attack 6), and mapping out critical URLs that get returned through error pages and modifying input used to generate the errors to inject SQL, XPATH, or XQUERY commands (attack 7). Once compromised, the adversary can use components to corrupt the target and/or exfiltrate information, including accessing sensitive data that should not be made available remotely and disseminating it to unauthorized external receiver endpoints.

This paper describes an innovative framework for remote monitoring that (1) strengthens overall security by limiting unintentional increase to the resulting attack surface and (2) can operate in contested network environments, including transient and high-latency network links. We argue that such a remote monitoring framework is a key enabler for the larger concepts of reactive and proactive cyber resiliency, as cyber decision making is inevitably driven by sensor information capturing the effects of both attacks and defender-initiated actions. The framework is currently being developed under the Gestalt project in support of DARPA's Integrated Cyber Analysis System (ICAS) [7] program. The objective of Gestalt is to provide federated access to a large diverse set of cyber observables to enable detection of targeted cyber attacks. Gestalt automatically discovers available data sources, unifies access to observables via a comprehensive common ontology, and automatically decomposes and federates queries and semantically integrates the results. The implementation status of framework is at a Technology Readiness Level (TRL) of 4, with basic functionality tested in the development environment.

The Gestalt system eliminates tedious manual inspection by providing access to all data sources on the network via a federated query interface. Using a new Cyber Defense Language, a single query can access data residing on multiple devices, across disparate device types and data formats, and return the query results in a semantically integrated and immediately useful format. Gestalt allows the cyber defender to focus on the forensic data itself by abstracting away the actual methods and techniques required to access that forensic data. Through its Semantic Query Decomposition capabilities, Gestalt infers the types of data sources that can be used to satisfy a given query, and identifies where instances of those data source types can be found on the network. Next, it dispatches native queries to the device containing each data-source instance to process the request. The results are semantically integrated and returned to the cyber defender. Gestalt provides a single interface to the cyber defender, dramatically improving their effectiveness and allowing them to focus their time and expertise on forensic analysis of the results of their search queries, rather than on the laborious process of data collection and processing.

The paper is organized as follows: Section II describes related work. Section III provides a high-level overview of the security architecture in relation to a threat model. Section IV dives into network-level security arguments while Section V provides more details on process-level security arguments. Section VI concludes the paper.

II. Related Work

The remote monitoring framework presented in this paper relates to work performed in cyber event monitoring, network monitoring, and stream processing/big data platforms.

A number of commercially available solutions exist in the Security Information and Event Management (SIEM) and cyber monitoring product space today, including ArcSight [8] and the Host Based Security System (HBSS) [9]. While Gestalt provides detailed access to the current system state, SIEMs provide extended summary information at a coarse granularity.

A number of solutions exist for network and grid monitoring [10], including Ganglia [11], Nagios [12], and Zabbix [13]. These systems specialize on performance monitoring and provide operators with dashboard views on the current availability state of the overall network system. Similar to Gestalt, many of these systems perform monitoring through a distributed set of nodes that report to a centralized monitoring dashboard. Ganglia in particular uses daemons installed on all monitored devices and meta-daemons that poll XML over TCP from daemons and meta-daemons. While similar, Gestalt is designed with a stronger focus on security rather than performance, including mandated use of TLS, no-listening sockets (analogous to meta-daemons), and process-level isolation.

Finally, a number of big data platforms exist for distributed processing of information. Splunk [14][15] is a well-known instance of a big data processing capability that makes it easy for cyber defenders to establish correlations between disconnected pieces of text information through a specialized query language. Unlike Gestalt, Splunk is based on an information model that requires raw observables to be aggregated in a central database before they can be queried. For instance, Splunk requires *forwarders* to be installed on data source devices, which establish connections outbound to indexers and communicate a large amount of data to those connections. In contrast, Gestalt reaches into data sources in a controlled way, leading to a much reduced attack surface.

III. Security Architecture

To better understand the security implications of adding systems like Gestalt to an already existing IT infrastructure, we

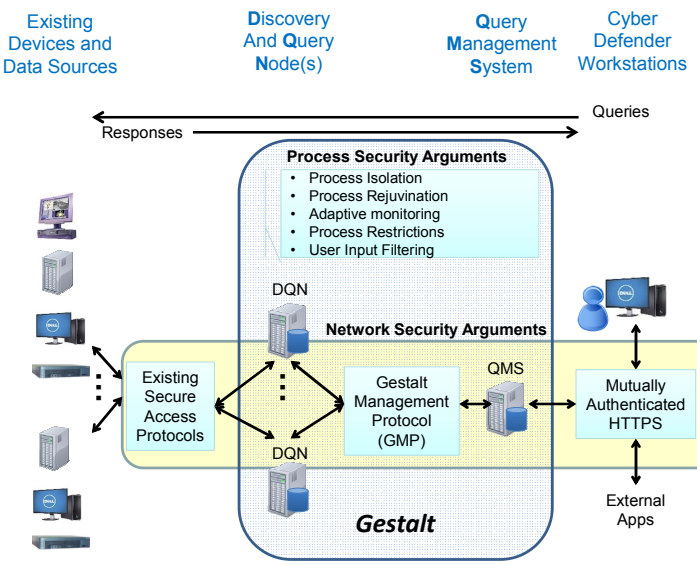

Fig. 2. Gestalt High-level Architecture

first look at the overall architecture of the resulting system. Fig. 2 shows the main components involved in remote monitoring performed by cyber defenders (on the right) of devices (on the left). The figure shows how the Gestalt system introduces two primary system components: the Discovery and Query Nodes (DQNs) and the Query Management Service (QMS).

The DQNs provide the interface between the devices on the network and the Gestalt system. Rather than adding new protocols to be implemented by end devices, the DQNs leverage existing protocols where possible (Fig. 2 left), including the Simple Network Management Protocol (SNMP) v3 [1], SSH, and the Distributed Management Task Force (DMTF) Web Services for Management (WS-MAN) [16] technologies, to communicate with individual devices and to catalog the data sources each manages. The QMS provide the main interface to cyber defenders and interacts with multiple DQNs through query dispatch and response interpretation logic.

The overall resiliency argument for this architecture consists of two main parts: network and process arguments. By constructing a sound network argument, we ensure integrity and confidentiality of the data transmitted over the network as well as a robust means for authenticating various actors, including processes and humans. The network argument is constructed using three different types of network protocols:

Device Access Protocols: In the cases where a single data source can be reached by multiple access protocols, the DQN will automatically select the strongest access protocol available by preferring authenticated and encrypted protocols. In addition, the DQN will enforce policy control over the choice of access protocols to be used to block use of protocols that are known to be unsafe and where the risk of using the protocol exceeds the benefits of getting observables from the data source through the protocol. For instance, the DQN might be configured to avoid communicating with TLS endpoints that are vulnerable to the Heartbleed [17] attack vector in order to protect the DQN's client process from compromise. Finally, to achieve availability, the DQN may fail over between acceptable access methods to achieve visibility in degraded mode.

Gestalt Management Protocol (GMP): The GMP specifies interactions between the QMS and the DQNs in a way that minimizes the attack surface in the DQNs and ensures operation in contested network environments through asynchronous polling semantics. Since this protocol is added to the existing system, it is constructed with strong security controls and principles in mind, including the use of TLS v1.2 [18] tied in with a robust PKI infrastructure, e.g., maintained by DISA for the DoD. Firewalls restrict allowable GMP communication to a dedicated QMS per DQN.

QMS Access Protocol: The QMS offers a RESTful [19] API that enables access by modern Web Browsers and external applications. Traffic is protected at the highest level supported by Web Browsers and external applications, e.g., currently TLS v1.2 for a selected subset of browsers. Firewalls restrict allowable communication to a set of known IP addresses.

The result of the network argument is a management protocol that can be added to existing IT infrastructure as a solid foundation for other resiliency techniques to build upon. The process argument for DQNs implements resiliency at the application-level through the following means.

Process Isolation: The DQN is split into two main components – a long-lived Manager process and multiple transient Bridge processes.

Process Rejuvenation: The Manager can temporally constrain the effects of compromised Bridge processes by killing the process as soon as its intended functionality is complete, where intended functionality can be defined over a set of requests.

Adaptive Monitoring: The Manager controls the lifecycle of Bridges by spawning them, observing their state, and killing them if they deviate from the norm. Anomaly detection is based on simple statistical methods, based on variance analysis of usage patterns on underlying resources.

Process Restrictions: Mandatory policy enforcement is enabled (e.g., using SELinux [20]) to explicitly allow the minimal set of interactions required for the process to function following a default-deny paradigm.

User Input Filtering: Any data resource from data sources is filtered and sanitized by transcribing it into a different representation format. Filtering includes checking maximum data size and sanitization includes turning the inputs from their native representation format into RDF/XML.

The process arguments for the QMS are similar and omitted from this paper for the sake of brevity.

IV. The Gestalt Management Protocol

The GMP specifies the connection behaviors and message exchanges between the QMS, the DQN Manager, and the DQN Bridge processes. The GMP was designed based on the following goals aimed at keeping the protocol vendor independent, secure, and implementable.

Type centric: GMP specifies the message structure rather than the programming. This enables the use of Transmission

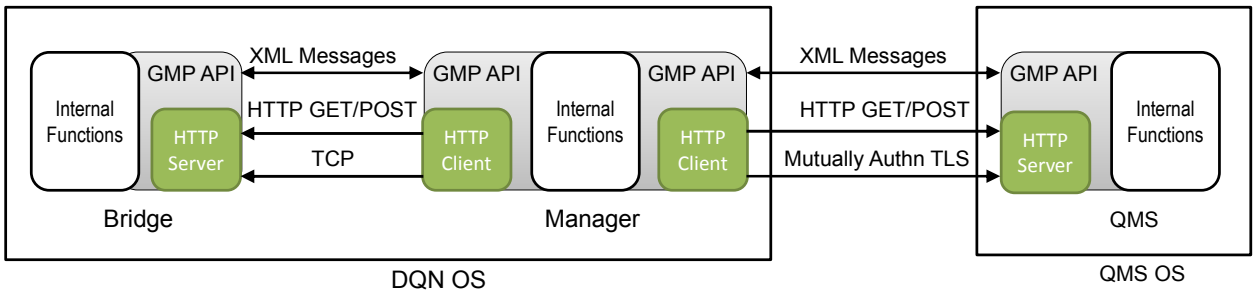

Fig. 3. The Gestalt Management Protocol (GMP) Architecture

Control Protocol (TCP)/IP-based load balancers and also achieves central processing unit (CPU) architecture, operating system, and programming language independence.

Network friendly: GMP works through firewalls that perform Network Address Translation (NAT) and optimizes network usage by virtue of using HTTP response codes, avoiding unnecessary round trip paths.

Secure: GMP uses a mutually authenticated Public Key Infrastructure (PKI) and TLS for encryption.

Pragmatic: GMP is designed to be implemented on top of a strong commercial and open-source community.

The overall architecture of GMP is displayed in Fig. 3. The figure shows a QMS process (QMS) on the right, a DQN Manager (MGR) process in the middle, and a DQN Bridge (BRI) process on the left. The Manager requests commands from the QMS and sends them to the Bridge processes for execution. The Manager checks for response messages from the Bridge and sends them to the QMS. The Bridge, Manager, and QMS implement a GMP-compliant interface in addition to their internal functions.

The GMP specification defines interactions between QMS, MGR, and BRI. As shown in Fig. 3, the QMS and BRI host HTTP servers, while the MGR incorporates a library/component that participates as an HTTP client. The MGR connects to the QMS by establishing a mutually authenticated TLS connection to the QMS's HTTP server. This methodology ensures the MGR is not exposed to direct network attacks because it does not provide a listening network socket. Listening network sockets open up a significant amount of kernel code to remote attack because packets need to be read in from the network and interpreted before authentication is performed. By switching to outbound-only connections, the MGR not only minimizes the resources set aside in their TCP/IP stack but also causes early failures if responses do not match up with requests. Also note that the communication between MGR and BRI is routed over the DQN's trusted loopback network. The Bridge must bind its HTTP server to the loopback network only. Therefore, the DQN does not provide any externally resolvable listening socket. Also, since the loopback network is trusted, the connection between the MGR and the BRI goes over plain TCP connections.

The reason for selecting HTTP between MGR and BRI over other local communication means, e.g., JAVA RMI, is that (1) HTTP does not introduce any other dependencies (such as registry components) that need to be secured and (2) also allows for dedicated connection per request interaction patterns that tradeoff increased security with performance. The reason for favoring a polling approach between MGR and QMS over push approach, e.g., Bidirectional-streams Over Synchronous HTTP (BOSH) [21], is as follows. First, polling with a connection per request model works better in disruptive environments, where constant network connectivity cannot be assumed, e.g., monitoring of cyber assets close to the tactical edge. Second, polling requests can be used as an active measurement of connectivity between the clients and the server (like a heartbeat protocol) to detect issues before data sharing is required without the need to add heartbeat protocols on top of HTTP. Third, limiting the lifespan of active connections leads to a reduced attack surface. Finally, the overhead associated with polling is small given the number of DQNs involved and the strategic use of HTTP 204 response codes. Conversely, requirements on end-to-end latencies for queries are to reduce access time from days to minutes, which make sacrificing some end-to-end latency for increased security and robustness worthwhile.

Note that if firewalls get in the way of allowing inbound connections to the QMS, a different optional GMP interaction pattern can be developed and deployed which includes a Gate-

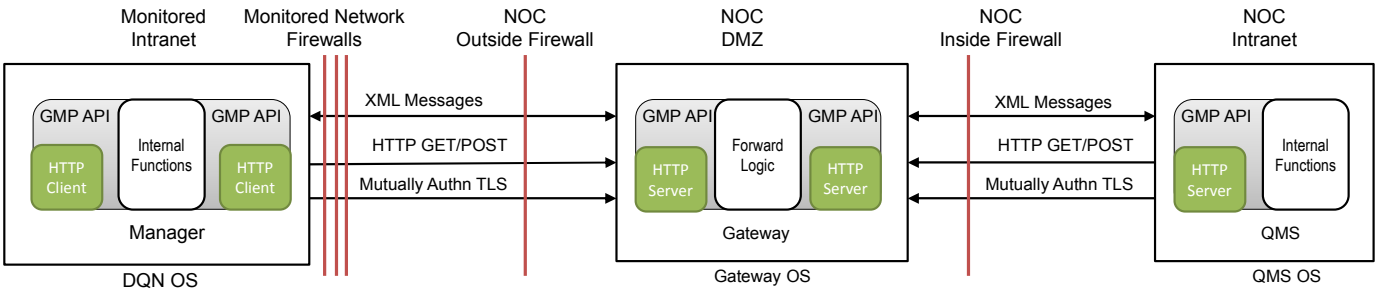

Fig. 4. GMP deployment in a Demilitarized Zone (DMZ) of a Network Operations Center (NOC)

TABLE I. GMP COMMANDS

Command	Description
getStatus()	Instructs the MGR to respond with a report that identifies the current runtime state of the MGR, including all of the runtime state of the BRI it manages.
abortCommand(id)	Abort command identified by id.
setConfiguration (configuration)	Apply configuration settings expressed in configuration XML document.
executeQuery (SPARQLquery)	Execute the query and return the results.
scheduleDiscovery (min, hour, month, dom, dow):	Sets up a schedule for running active discovery, one time or repeating. Note that the scheduling is similar to the Linux or UNIX "cron" command.

TABLE II. GMP RESPONSES

Response	Description
Ack	Acknowledging the MGR has parsed a given commandMessage and will take action per *the ackResponse* attribute. The ackResponse is exactly one of: *Working*, *Invalid*, or *Inappropriate*. *Working* indicates that the MGR is about to begin processing the specified command block. *Invalid* indicates one or more parsing errors in the commandMessage. *Inappropriate* indicates that one or more commands in the commandMessage are inappropriate for that MGR, e.g., unsupported commands.
Nack	When the MGR receives a commandMessage it is unable to parse to retrieve a commandMessage@ID, a Nack message is returned.
Success	The MGR acknowledges that a given set of commands successfully executed. The results is one of (1) nothing, (2) a statusReport, or (3) a queryResult.
Failure	The MGR notifies the QMS that a given commandMessage failed to successfully process. Failure types supported include • unreachableError: Reachability problems to components critical for command execution • storageExceededErrorType: Problems with persistence store overrun on the DQN • invalidStateErrorType: Problems with exceptions triggered by command • interruptedErrorType: Problems with commands timing out • malformedContentErrorType: Problems with parsing data supplied by the QMS

way process placed into the QMS's Demilitarized Zone (DMZ). This Gateways process, shown in Fig. 4, allows both the DQN and the QMS to make outbound-only connections through their respective firewalls. The Gateway logic is simply to forward GMP messages between the two endpoints. To prevent against corruption of the Gateway due to its exposed location in the DMZ, GMP messages may optionally be protected via signatures implemented through XML Signature and encrypted using XML Encryption.

A. Connection Management

Communication between MGR and the QMS must be via HTTPS. In order to reduce the attack risks to the MGR, the following constraints apply:

- All communications must be initiated by the MGR, never the QMS. With respect to GMP, the MGR is a client only and never a server.
- Both the MGR and QMS mutually authenticate all communications via mutually authenticated certificate exchange.
- Both the MGR and QMS must ensure that the certificates of the other device have not been revoked or expired.

The following interaction patterns are designed to optimize network usage:

- For polling intervals in the minute range, TLS connections are created in a dedicated manner for the poll request. For short lived intervals (seconds), TLS connections may be reused.
- The QMS returns HTTP response 204 (no content) in case no commands are available

B. Command Management

If the MGR polls the commandURI and receives an HTTP 204 message, there are no commands for that MGR at that time and the cycle is complete until the next polling interval.

Linking responses to commands over multiple HTTP connections is maintained by repeated use of a UUID attached to the initial commandMessage. This UUID is referred to as the commandMessage@ID and is referenced in each responseMessage sent from the MGR to the QMS via the root_id element. The one exception to the commandMessage@ID being equal to responseMessage@root_id is when a nack response is sent, informing the QMS that the MGR was unable to parse the content. Note that the responseMessage contains another optional id element, called part_id, which is used for two purposes. First, some commands, like the scheduleDiscovery command, cause asynchronous processing to happen multiple times, with a responseMessage being generated each time. In this case, the root_id will point to the commandMessage in which the scheduleDiscovery command was included, and each responseMessage will have a unique part_id. The second

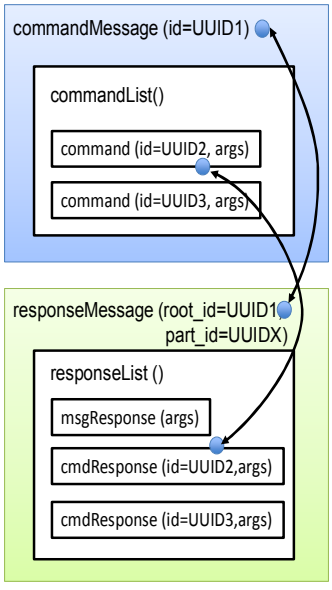

Fig. 5. Message Overview

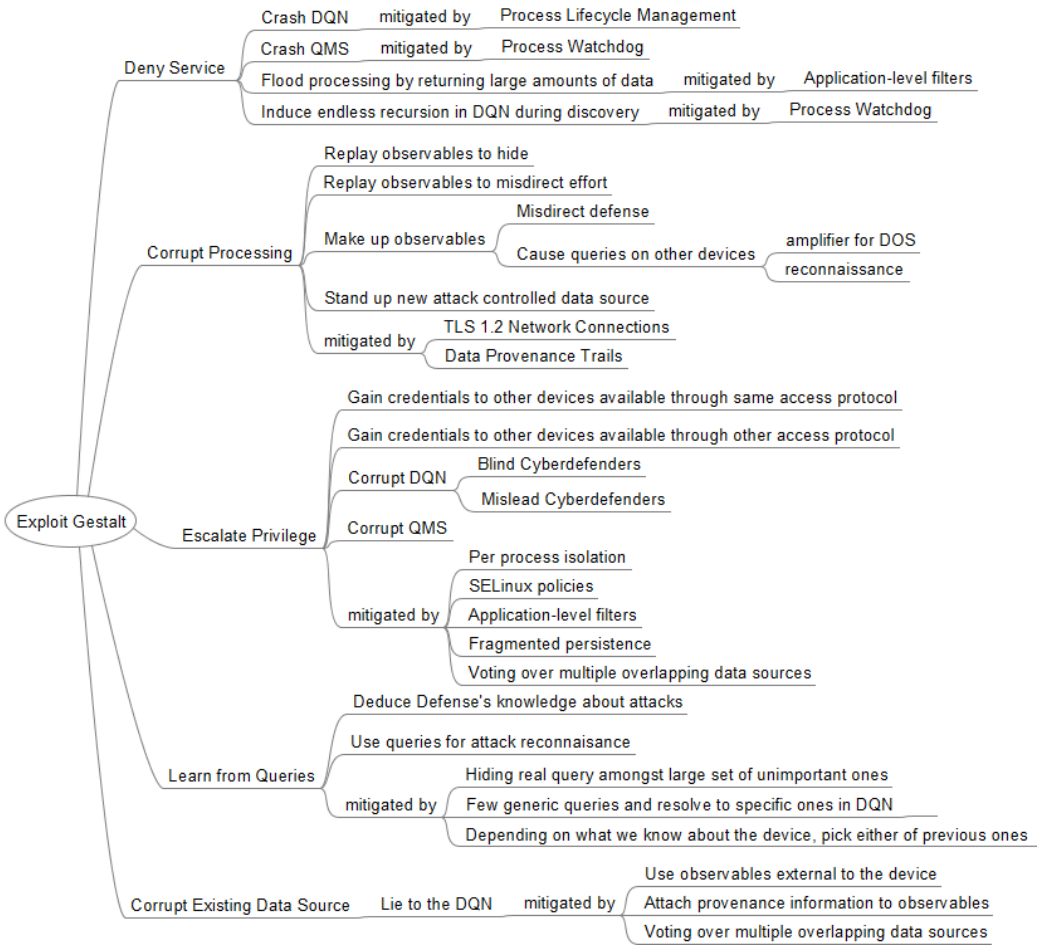

Fig. 7. Threat Model

use case involves a single query and response exchange in which multiple reponseMessages are returned, each one containing a partial result set. Again, the root_id will point to the id attribute of the commandMessage carrying the query, while the part_id attribute is unique to the specific responseMessage.

Similarly, each response chosen by the MGR to send to the QMS is wrapped in a responseList. The MGR sends the responseMessage via HTTP POST to the responseURI of the QMS. The same way the responseMessages are linked to commandMessage at the message level (through shared use of id values), responses and linked to commands at the command level through shared use of id attributes. Graphically, these XML structures can be depicted as shown in Fig. 5.

TABLE I lists the current set of commands, and TABLE II lists the current set of responses.

V. Process Security Arguments

Given the strong security guarantees provided by network protocols described earlier, the next level of constructing a resiliency argument deals with a threat model in which the adversary has compromised one of the data sources that is being accessed by the DQN, see Fig. 6.

For the purpose of evaluating potential threats and designing mitigations, we created a threat model in the form of an attack tree, shown in Fig. 7. From left to right, the attack tree decomposes the high-level goal of "Exploiting Gestalt" into five specific attack branches as follows:

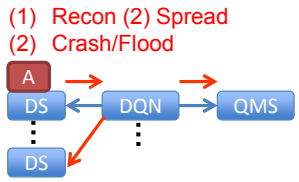

Fig. 6. Starting Point: Corrupted Data Source

- Learn from Queries, causing loss of confidentiality by misusing queries issued by Gestalt for attack reconnaissance.
- Escalate Privilege, causing loss of integrity in various Gestalt components and loss of confidentiality for access credentials in the process.
- Deny Service, causing loss of availability.
- Corrupt Data Source, causing loss of integrity of analysis results by generating bad observables on the locally compromised data source.
- Corrupt Processing, causing loss of integrity of analysis results by externally spoofing data source observables.

Each branch is further refined into specific attacks. In addition,

we annotate mitigation mechanisms towards the leaves via specialized "mitigated by" branches.

A. Learn from Queries

Starting from a compromised monitored device, an adversary observes specific queries issued by the DQN against that device. For example, if the DQN issues commands against binaries looking for a specific string S, the adversary knows that attack binaries containing S will be detected easily going forward and hence it would be prudent not to use S anymore if the objective is to stay dormant. Even worse, the query contains information containing other sensitive devices on the network, e.g., show me all outbound connections from end systems to a highly sensitive server. In this case, the adversary might learn information about the highly sensitive server with only sufficient privileges on a Gestalt-monitored edge device. In some sense, Gestalt would do reconnaissance on behalf of the adversary in this case.

While the data aggregation problem in very hard to solve in general (e.g., two pieces of unclassified information becoming highly classified), Gestalt mitigates against such attacks through the following means:

1. The DQN can send a potentially large number of queries to the device, hiding the real query among the set of all queries. The benefit of hiding needs to be carefully balanced with the increased load on the network and end system.

2. The DQN can send a more general query to the monitored device, and then subset the responses internal to the DQN. This pulls out more data from the actual device into the DQN, which might expose it a little more through centralization (something that ICAS is explicitly trying to avoid).

Specific choices can be customized based on what is known by the DQN about the monitored device before it is actually accessed. For instance, if a device looks suspicious based on its network behavior, it might make sense to employ either strategy 1 or 2 for this specific device, while maintaining a default behavior of sending specific queries to devices otherwise.

B. Escalate Privilege

Since we assume that monitored devices are compromised and the DQN needs to interact with those devices, there is a clear path for adversaries to spread out to other devices through Gestalt. One attack would be to gain access to a larger set of credentials used in the DQN to gain remote access to other devices, either through the same access protocol or different access protocols. This can be achieved, for instance, by sending some input to the data interpreter on the DQN that causes a buffer overflow. Along the same lines, attacks might directly target the DQN functionality and metadata information to either blind cyber defenders or to misdirect efforts. Finally, the attacker's next logical step would be escalating privileges to the QMS, and from there, compromising the browser used by cyber defenders to access the QMS.

Gestalt mitigates these attacks using a strong containment strategy throughout the DQN, isolating adapters that interact directly with the devices from extractors and management components. Isolation is done on a per process basis with custom SELinux policies written for each process restricting access down to individual files and network resources. Persistent storage is also fragmented into per process files (e.g., key stores), a metadata index database, and a configuration store. Finally, interactions between components are controlled by application-level filters for user input validation. Corruption is limited by process rejuvenation techniques, e.g., starting new processes and limiting the amount of time they are allowed to linger before disappearing after not being used. Depending on the level of security paranoia required to interact with devices, the following configurations for adapter processes:

- Per device adapter processes, with dedicated access to only the credentials available to access that device,
- Per domain adapter processes, where a domain can be a grouping over devices along administrative domains,
- Per protocol adapter processes, where all SNMP devices are handled through a single SNMP adapter instance with a single credential file.

Finally, the DQN host itself is installed and administered using best-practice secure host techniques, exposes only those services absolutely necessary for the operation and maintenance of the system, and uses only the highest level of security applicable for those services (i.e., Public Key credentials for authentication, encrypted network protocols, IP source address filtering for new connections, and so forth.)

C. Deny Service

With the specific attack objective to cause loss of availability, adversaries might crash various main components (DQN, QMS) or sub-components of them (adapter processes). Another way to deny service is to overload shared resources, e.g., return a large amount of data to the DQN to cause out-of-memory exceptions. A more intricate version of causing denial on the discovery component of the DQN is to create conditions that cause the discovery algorithm to spin out of control, e.g., by creating unexpected loops in observables.

Gestalt mitigates these attacks by explicitly and actively managing the lifecycle of processes and jobs that these processes need to perform. Processes are monitored for activity and functionality, and information about problems is relayed back from the DQN to the QMS. Query and discovery jobs are explicitly tracked through IDs and can be started, stopped, and referred to for fetching completion results. These jobs will also be granted access to only those resources (disk, memory, etc.) deemed necessary for their functioning and will be terminated before a resource exhaustion issue could affect the operation of the overall system.

D. Corrupt Data Source

This part of the attack tree captures the fact that devices can lie to the DQN if the reporting mechanism used by the device is also corrupted (which is generally assumed). Put this way, the benefit of using a DQN to access state reported by the corrupted device itself needs to be treated differently from information obtained from other devices (e.g., NIDS) about that device.

Gestalt mitigates these attacks by keeping track of provenance information associated with observables to enable high-

er-level reasoning and deconfliction by cyber defenders. Provenance information is assembled by various components of Gestalt, including the DQN and the QMS, in a crypto-strong way that allows for integrity checks in the QMS.

E. Corrupt Processing

The overall goal of this attack is to use Gestalt against the defender by corrupting its functionality in various ways. For instance, traffic replay (at various points, not just the device access protocols) might make it look to Gestalt as if the world either did not change (although it changed) or did just significantly (although it did not). The goal of the first case is to hide within the Gestalt monitoring framework, while the second case is to cause cyber defenders to go down rat holes that take focus and attention away from the real issues at hand. An interesting case involves corruption in the form that causes the Gestalt operators to actively do work on behalf of the adversary. If a DQN is corrupted, can it, by reporting certain values back to the QMS, get the defenders to execute queries on other DQNs and then feed back the results of those queries into the corrupted DQN? Such a setup could also be used for an amplified denial-of-service attack, where the results reported back from a single corrupted DQN could warrant a wide range of QMS initiated interactions with other DQNs and devices.

VI. Conclusion and Next Steps

The current enterprise IT infrastructure remains vulnerable to targeted cyber attacks that stay undetected for months, despite the fact that a variety of low-level observables are available, audited, and recorded. This establishes the need for a remote monitoring framework that can integrate with existing data sources in a secure manner, dispatch queries from a unified presentation to specific data sources at hand, and securely integrate results back into a consistent and reliable cyber operational picture.

This paper describes the security and resiliency architecture for the remote monitoring framework we are currently developing under the DARPA ICAS program. It describes how this framework strategically combines strong network resiliency and protection with process-level resiliency techniques, including isolation, rejuvenation, and adaptive monitoring/response. The overall claim is designing security and resiliency into the architecture from the beginning and in a bottom-up way allows creation of the argument that the value of adding the new monitoring framework to an already existing IT infrastructure outweighs the risks associated with increasing the attack surface.

Going forward, we plan to participate in an external evaluation of the security argument performed by an adversarial partner as part of the ICAS program. In addition, we plan to provide implementations for an increasing set of mitigations outlined in the attack tree, and further refine the tree by adding linking in integration tests via "verified by" leaf branches.

References

[1] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[2] J. Rosenberg and D. Remy, *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Pearson Higher Education, 2004.

[3] OMG, "Common Secure Interoperability Protocol Version 2," Mar-2002. [Online]. Available: http://www.omg.org/spec/SEC/1.8/PDF/.

[4] W. M. Eddy, "TCP SYN flooding attacks and common mitigations," 2007.

[5] M. Atighetchi and J. Loyall, "Meaningful and flexible survivability assessments: approach and practice," *CrossTalk- J. Def. Softw. Eng.*, vol. 23, no. 2, pp. 13–18, 2010.

[6] M. Schoenefeld, "Pentesting J2EE," *Blackhat*, 2006. [Online]. Available: http://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Schoenefeld-up.pdf.

[7] DARPA, "Integrated Cyber Analysis System (ICAS) Homepage," 2014. [Online]. Available: http://www.darpa.mil/Our_Work/I2O/Programs/Integrated_Cyber_Analysis_System_%28ICAS%29.aspx.

[8] D. Miller and B. Pearson, *Security information and event management (SIEM) implementation*. McGraw-Hill, 2011.

[9] DISA, "The Host Based Security System," 2012. [Online]. Available: http://www.disa.mil/Services/Information-Assurance/HBS/HBSS.

[10] S. Zanikolas and R. Sakellariou, "A taxonomy of grid monitoring systems," *Future Gener. Comput. Syst.*, vol. 21, no. 1, pp. 163–188, 2005.

[11] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Comput.*, vol. 30, no. 7, pp. 817–840, 2004.

[12] W. Barth, *Nagios: System-und Netzwerk-Monitoring*. No Starch Press, 2008.

[13] R. Olups, *Zabbix 1.8 network monitoring*. Packt Publishing, 2010.

[14] J. Stearley, S. Corwell, and K. Lord, "Bridging the gaps: joining information sources with Splunk," in *Proceedings of the 2010 workshop on Managing systems via log analysis and machine learning techniques*, 2010, pp. 8–8.

[15] Splunk.com, "Splunk for Security – Supporting a Big Data Approach for Security Intelligence," 2014. [Online]. Available: http://www.splunk.com/web_assets/pdfs/secure/Splunk_for_Security.pdf.

[16] DMTF, "Web Services for Management (WS-MAN) Specification," DSP0226, 2010.

[17] Mitre CVE, "CVE-2014-0160," *Heartbleed Vulnerability*, 03-Jul-2014. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160.

[18] T. Dierks, "The transport layer security (TLS) protocol version 1.2," 2008.

[19] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, 2000.

[20] S. Smalley, C. Vance, and W. Salamon, "Implementing SELinux as a Linux security module," *NAI Labs Rep.*, vol. 1, p. 43, 2001.

[21] I. Paterson, D. Smith, P. Saint-Andre, and J. Moffitt, "Bidirectional-streams over synchronous http (bosh)," 2010.